REMARKS

Claims 1-7, 9, 11, 13-23, and 28-31 are presented for consideration. Claims 7, 9, 15, 16, 20, 28, 29 are currently amended. Claims 8, 10, 12, 24-27 were previously cancelled.

Claims 1-7, 9, 11, 13-23 and 28-31 are rejected under 35 U.S.C. §112, first paragraph, as failing to comply with the written description requirement. Specifically in reference to claims 1, 15, 20, and like dependent claims, the Office Action explains that these claims recite a control module having greater access permission to the client device than the network client application. The Office Action sees the level of access permission of the control module as new matter that is not supported in the specification. The Office Action explains that although the specification states that less security restrictions are placed on the stand-alone application, the Office Action believes that the specification is silent as to any access permissions of the control module.

Applications respectfully disagree. The specification explains that a network client application (i.e. web browser, and any applet it may launch) does not have access to the native functionality of the system, i.e., platform, on which the applet (or web browser) is being run (page 1, lines 12-22). In other words, a network client application cannot launch a stand-alone application on the system on which it is running (due to restricted access permissions). Consequently, the present invention addresses the question of how to use a network client application to initiate the launching of a stand-alone application that runs outside of the network client application.

The present invention addresses this issue by advantageous use of a downloadable control module. As is recited in the claims, and explained at least in page 13, lines 7-9 of the Specification, the download package includes installation files (INF files) that include a custom control module (OCX). This control module is granted greater access to the native platform than the network client application (i.e. is granted a secure operating environment for launching the stand-alone application) by use of a digital signature, or use of a dialog box through which a user may acknowledge (i.e. permit) installation of the control module, or by use of

public key cryptography techniques to securely identify the control module prior to it being allowed to run on the client machine (page 16, line 1 to 9).

As it is known in the art, these control modules are small autonomous programs (precompiled in machine code) that upon receiving permission, will be downloaded, installed, and coupled to the web browser. Once installed, these control modules have greater access to the client device than does the general web browser, i.e. the network client application. These control modules go under different names depending on the type of network client application they are designed to interact with (i.e. designed to be couple to). Examples of such executable control module programs are ActiveX control modules in Internet Explorer™ and Plugin control modules in Netscape™ (page 9, lines 3-11). Typically, these types of control modules are used to access lower level controls of a system platform (i.e. the client device) that would otherwise be restricted from the network client device. For example, control modules may access lower resources on a video card in order to enhance video images on a network client application window. The present invention, however, takes advantage of the control module's greater access permission to the client device to launch a separate and completely independent application on the client device (i.e. the stand-alone application described in the specification and claims).

Thus, Applicant believes that the specification clearly explains that the control module has greater access permission to the client device than the network client application, explains how the control module gains this greater permission (i.e. through an acknowledgment sequence), and provides an example of the greater control (i.e. the ability to launch a stand-alone application native on the client device and outside of the network client application).

In reference to "claims 9, 15, 20 and like dependent claims", the Office Action notes that these claims recite that the application includes a plurality of "user-available function tools". The Office Action sees the use of user-available function tools as new matter that is not supported in the specification. Specifically, the Office Action notes that the specification states the use of a plurality of application options or parameters, but asserts that nowhere in the specification do these options or parameters include user-available function tools.

Applicant concedes that the words "user-available function tools" are not explicitly recited in the specification, but remind the Examiner that it is not required that the claim language exactly match the wording used in the specification. It suffices that the recited claim material be supported by the specification. As was explained in a previous Office Action Response, it became apparent to Applicant that the Examiner had chosen to interpret the word "parameter" very broadly, to encompass more than what was envision by Applicant. Specifically, Applicant had used the words "options or parameters" to mean software tool selection (such as selecting from among a greeting card creation software tool, a business card creation software tool, a slide show presentation, and such). However, previous Office Actions chose to interpret the word "parameter" to apply to environmental parameters of a running application, such as the size of the window in which the application runs, the type of font available, etc. To avoid the unintended ambiguity that led to the expanded interpretation of Applicant's claim, it became necessary to avoid words that might be misinterpreted and substitute language having a more precise meaning in the art.

As is shown in Figs. 1 and 2, and explain in page 5, line 7, a user is provided with various application-type options, from which the user may select. Once selected, the independent application is configured to provide the selected application-type options (page 5, lines 13-15). The term "option" is sometimes substituted with the term "parameter", (page 4, lines 19-20; page 5 lines 18-20; page 5 line 24 to page 6, line 2). In either case, however, the user-selected options/parameters are defined as being a specific type of application tool, such as software tools for creating greeting cards, creating business cards, creating a slide show presentation, etc. (page 11, lines 6-21, and page 11 line 25 to page 12 line 3). The term software tool can be used in the art to refer to an application designed for a specific task, or function. Thus, Applicants had selected the term "user-available function tools" to explicitly recite that the options being made available to the user refer to specific application-type options.

However, Applicants concede that perhaps the term "function tool" is not precise enough. Thus applicants propose amending the claims once more to replace "user-available function tools" with "user-available application options" in order to

better recite that the options define different types of application functions (i.e. define application options for creating greeting cards, application options for creating business cards, or application options for creating a slide show presentation). Claims 9, 15, 16, 20, 28, and 29 have been amended accordingly.

Claim 7 was rejected under 35 U.S.C. § 112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicants regards as the invention. Specifically, the Office Action first notes that claim 7 recites in lines 1-3, "wherein selecting user-available function tools of the stand-alone application further includes ...". The Office Action then notes that, "there is insufficient antecedent basis for this statement in the claim. Nowhere in claim 7, or in claim 1 from which it depends, is there previous mention of a step for selecting user-available function tools of the stand-alone application ...". Applicants thank the Examiner for noting this oversight, and have amended claim 7 to remove mention of selecting user-available function tools.

Claims 1-7, 28, 29, and 31 are further rejected under 35 U.S.C. § 103(a) as being unpatentable over Spyker et al. (U.S. Pat. 6,571,389) in view of Grate et al. (U.S. Pat. 5,956,483). Specifically, the Office Action concedes that Spyker did not explicitly disclose using a network client application on the client device including the feature wherein the stand-alone application is launched on the client device outside of the network client application. However, the Office Action then declares that method for launching local (stand-alone) applications *from* a browser were well known in the art as evidenced by Grate. Applicants respectfully disagree. As Applicants explained in Applicant's Office Action response of June 5, 2006 (page 10, 4th paragraph),

> " Grate explains a manner by which a web browser may communicate with a stand-alone application running outside of the web browser. But Grate does not teach or suggest that the web browser launches the stand-alone application outside of the web browser. Indeed, Grate is silent on how his stand-alone application is launched. Thus, one must assume that Grate's stand-alone application is launched in the typical manner of clicking on a icon or using a command line execution."

Applicants contacted the Examiner for clarification of where Grate described that the application was launched from within the web browser. The Examiner noted that in col. 2, lines 39-49 of the Summary section states,

> "When the user initiates the function call, the HTTP POST message is delivered by the Local Host service to the port listener; the port listener then launches the main program module of the specialized application (if not already running), and passes the function calling information to the program module for execution".

Grate further elaborates on this in the Best Mode section, col. 12, lines 37-41, wherein it states,

> "With reference to Event D, the port listener 132B then launches the core code module of the Shopper 132 (if not already running), which runs as a separate process from the port listener 132B".

Applicants respectfully point out that both of these excerpts state that the "port listener", not the web browser, launches the stand-alone (i.e. the core module of "the specialized application"). That is, the port listener is not part of the web browser, and thus it cannot be construed that the stand-alone application is launched by the web browser. Grate explains a bit more about the port listener in col. 12, lines 55-62, wherein it is explained that in order to function, the port listener needs to already be running prior to the web browser starting. Specifically, Grate explains that,

> "The port listener 132B is implemented as a background process which runs independently from the Shopper 132 process, without showing any window on the desktop. To ensure that the port listener 132B is running when a L-WFCP function call is initiated by the user, the port listener 132B and operating system are preferably configured such that the port listener 132B is automatically launched during operating system start-up".

From the above, it is clear that the port listener is not part of the web browser, and since it is the port listener, not that web browser, that launches the program, it cannot be construed that Grate teaches that the web browser launches the stand-alone program (i.e. that the stand-alone program is launched from within the web browser).

Indeed, even if Grate did suggest that the web browser may launch a stand alone program, Grate does not provide any instruction for how such action may be implemented. That is, Grate does not explain how the web browser may overcome its limited access to the native machine's lower level operation in order to install and launch an independent program for operation outside of the web browser.

The Office Action further concedes concerning claim 29 that Spyker,

> "...did not explicitly disclose tools for creating greeting cards, creating business card, and creating a slide show presentation. However, software for creating greeting cards, creating business cards, and creating a slide show presentation was well known in the art at the time of the applicant's invention. Any application for creating greeting cards, creating business cards, and creating a slide show presentation would utilize various application options that would be specific to the application and that would allow for such creating. Since Spyker sets forth various properties/dependencies of an application, it is clear that these properties/dependencies would include tools for creating cards, creating business cards, and creating a slide show presentation when the application is an application for creating greeting cards, business cards, and creating a slide show presentation. Thus it would have been obvious to one of ordinary skill in the art ...to modify the system of Spyker by adding the ability to utilize tools for creating greeting cards, business cards, and creating a slide show presentation."

Applicants respectfully disagree. Applicants have previously explained that Spyker explains how to create a browser environment needed for running a JAVA applet without launching a browser. As Spyker explains, a stand-alone JAVA application cannot be launched from a web browser, but a JAVA applet can. However, the Java applet is limited to running within the web browser. Thus, Spyker explains how to collect all the environmental parameters (i.e. operating system settings and JAVA environment settings) needed for simulating a web browser environment in which a Java applet may be launched, then encapsulating these parameter settings into a launcher application that can launch the Java applet. The point is the Spyker is not concern with what operation the applet may execute (i.e. it is irrelevant whether the applicant may be used for creating greeting cards, creating business cards, or creating a slide show presentation) since all applets will likely require the same environmental parameters (i.e. those parameters needed for launching and running an applet...any kind of applet).

As applicants have repeatedly tried to explain, the present invention is not concerned with the operating environment of the stand-alone application, and the term "application options" as used in the claims (please note that the term "parameter" does not appear anywhere in the current claims) refers to the specific type of operation the application will execute. The present application is capable of multiple different tasks (these multiple tasks share a common software engine), but can cater its interface to facilitate the execution of any one desired task if the other tasks will not be needed. As the claims recite, once a user has selected a specific application option, the stand-alone application configures itself to include the selected application options and to *exclude* the unselected application options from among its plurality of application options.

Even if one were to equate Spyker's operating environment parameters with the task operations of an application, one would still not achieve the present invention since the present invention requires *including* and *excluding* specific application options from launched application based on a user's *prior* submission of his/her *intended use* of the application.

Furthermore, since Spyker explicitly describes a method of creating an operating environment for launching a JAVA applet specifically to avoid launching a stand-alone JAVA application (which Spyker explains is difficult and not possible from within a browser), there is no incentive for combining the teachings of Spyker with any teaching of launching a stand-alone JAVA application from a browser since such an operation is directly opposed to Spyker's teachings and would render his invention inoperable for its intended use. That is, if one could launch a JAVA application from a browser for execution outside of the browser, there would be no incentive for creating an artificial browser environment for launching a JAVA applet, which is Spyker's goal.

This Response After Final Rejection is believed to place this application in condition for allowance and its entry is therefore believed proper Under 37 CFR §1.116. At the very least, however, it is believed that the formal rejections under 35 U.S.C. §112 have been overcome. Accordingly, entry of this Response After Final Rejection, as an earnest attempt to advance prosecution and reduce the number of issues, is respectfully requested. Should the Examiner believe that

issues remain outstanding, he/she is respectfully requested to contact applicants' undersigned attorney in an effort to resolve such issues and advance the case to issue.

Respectfully submitted,

/Rosalio Haro/
Rosalio Haro
Registration No. 42,633

Please address all correspondence to:

Epson Research and Development, Inc.
Intellectual Property Department
2580 Orchard Parkway, Suite 225
San Jose, CA 95131
Phone: (408) 952-6131
Facsimile: (408) 954-9058
Customer No. 20178

Date: November 22, 2006